Reswitching of Connection Networks

By M. C. PAULL

(Manuscript received November 28, 1961.)

In certain types of connection networks, it is always possible to unblock a blocked call by moving calls already set up in the network. The following results relating to these networks are derived in this article.

1. Bounds on the number of calls which must be disturbed to unblock a blocked call.

2. Bounds on the relation between the number of calls which are already set up in the network, and the number of calls that must be disturbed to unblock a blocked call.

3. Methods of systematically changing connections to unblock a blocked call.

1. INTRODUCTION

In a three-stage network of the type pictured in Fig. 1, it is possible that a connection between an input and an output cannot be made despite the fact that neither is already connected. This could happen if other connections already occupy at least one link in every possible path between the input and output in question. As first established by Slepian,[1] a blocked connection in such a network can be unblocked by rearranging the connections already set up in the network. Slepian further showed that such a rearrangement would never require disturbing more than $2n - 2$ calls, where the size of the switches in each stage is n by n, and there are n switches per stage. In the first sections of this article I give a proof that *to unblock a connection in such a network in no case requires disturbing more than* $n - 1$ *calls, and furthermore for every* $n > 1$ *there is at least one network state in which* $n - 1$ *calls must be disturbed to unblock a blocked connection.*

In subsequent sections various generalizations upon which partial results have been obtained are discussed. These include results on different network configurations, and networks having more than three stages.

As discussed in the body of this paper, the physical consequence of a

network change is to momentarily disturb network connections — to open some of these connections during the time that changes are being made. Depending on the application of the network, and the time duration of the disturbance caused by the change, this disturbance may or may not be of serious consequence. In an electromechanically operated network used to connect telephone calls such disturbances might result in disturbing conversations carried by the network. Fortunately, one can design switching networks and find change algorithms for such networks such that there will be no such disturbances. In the Appendix such a network and algorithm is described.

II. MATHEMATICAL MODEL

2.1 *The Network*

We first need a mathematical model of the network of Fig. 1 (which will simply be called "the network" from now on) in which we may conveniently represent the possible states of the network, and in which the basic properties of this particular type of network are made exact.

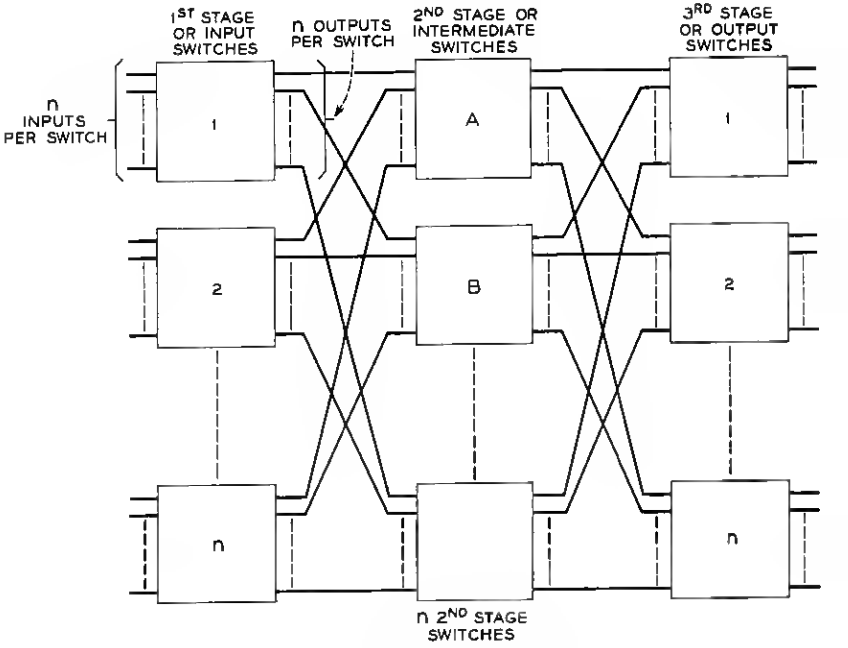

Fig. 1 — Three-stage network suitable for reswitching.

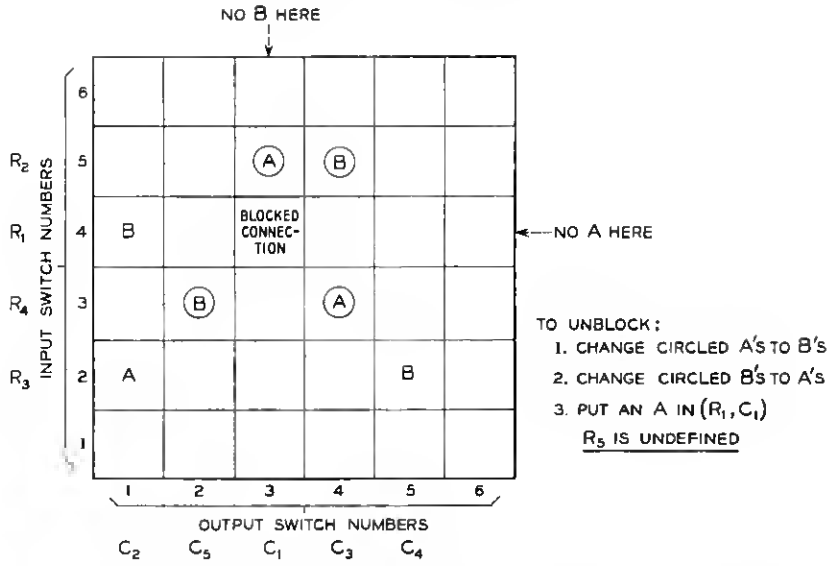

Fig. 2 — Matrix for representing the state of a three-stage network. The state pictured in the figure is illustrative of a typical blocked state as discussed in the sufficiency part of theorem 1.

We will represent the connections existing in a given network state by a set of symbols entered in a matrix (Fig. 2). The matrix has n columns and n rows, and there are n possible symbols which may be placed in any matrix position. Each position may contain from zero to n symbols. The n rows correspond to the n-input (first stage) switches; these are numbered $1, 2 \cdots n$. The n columns correspond to the n-output (third stage) switches, and these are numbered $1, 2 \cdots n$. The n symbols correspond to the n intermediate (second stage) switches. To indicate a position in the matrix we use the ordered pair (a, b), where a is the row and b is the column. An entry, say Q in matrix position (a, b), corresponds to a connection from input switch a through intermediate switch Q to output switch b. No entry in (a, b) indicates the absence of any connection from a to b. Although the matrix entry does not indicate which input line is connected to which output line, it does uniquely specify the links (links are the nodes in which first and second, and second and third stage crosspoints meet) involved in such a connection. For our purposes this is the important property of a connection from an input to an output line.

There are certain restrictions on the set of connections (network state)

which can exist in a network of the type of Fig. 1. These must he reflected in restrictions on the set of entries possible in our matrix:

i. There can be no more than n symhols in any row or column. This corresponds to the fact that there only n inputs to each input switch and only n outputs from each output switch.

ii. No two symhols in any row (column) may he the same. This corresponds to the fact that each input (output) switch has only one connection to each intermediate switch. If the same symhol appears more than once in a row or column, the different appearances of the symhol will he said to conflict.

A matrix with entries meeting the ahove restrictions will be called "legitimate," or the entries will be called "legitimate."

2.2 *Blocking-Unblocking*

Given a matrix with a set of entries, corresponding to a network having a corresponding set of connections, it may be impossihle to make an entry in (a, b) and still have a legitimate matrix. This corresponds to the impossihility of setting up an additional conneetion between input switch a and output switch b. The two possihle reasons for such an occurrence are:

i. There are already n symhols in row a or n symbols in column b.

ii. There are already a total of n different symbols in row a and column b, but there are less than n symbols in row a, and less than n symbols in column b.

If *i.* holds, (a, b) will be said to be trivially blocked. This corresponds to the ease where either all input lines to input switch a or all output lines from switch b, or both are already connected.

If *ii.* holds, (a, b) will said to be blocked, or legitimately, or non-trivially blocked. This corresponds to the case in which an input line on switch a cannot be connected to an output line on switch b despite the fact that neither is already connected.

Note we do not have to be specific about input and output lines, hecause a connection hetween an input and output line is legitimately blocked if and only if their corresponding switches are legitimatcly blocked. This is so hecause all switches in the network are non-hlocking.

We will speak ahout changing connections of a network in a given state. By this we will *not* mean changing the input and output switch involved in the connection, but changing only the intermediate switch involved. That is, if a network has a connection between a certain input line and a certain output line before a *change*, it will still have a connection from the input to output line in question after the change. In terms

of our matrix a *change* corresponds to changing the symbols at various positions, but a change leaves the number of symbols in any position unchanged. A legitimate change is one which does not result in a matrix (set of network connections) which violate restrictions *i.* or *ii.* of Section 2.1.

By unblocking a blocked connection (a, b), we mean making legitimate changes in matrix symbols (network connections) in such a manner as to provide that there are a total of at most $n - 1$ different symbols in row a and column b. In the sequel we prove a theorem on the maximum number of such changes which are sufficient and necessary to unblock any connection.

2.3 *Theorem 1*

In order to unblock a blocked connection in a network, no more than $n - 1$ changes are required. For any $n > 1$, there are network states in which a connection is blocked that require $n - 1$ changes to unblock that connection.

Proof. Figures 2 and 3 are provided to aid the reader (and the author) in following the proof.

2.3.1 *Sufficiency*

Assume (r_1, c_1) is non-trivially blocked. This implies that *there is a symbol, say A, in column c_1 which does not appear in row r_1*. Because if

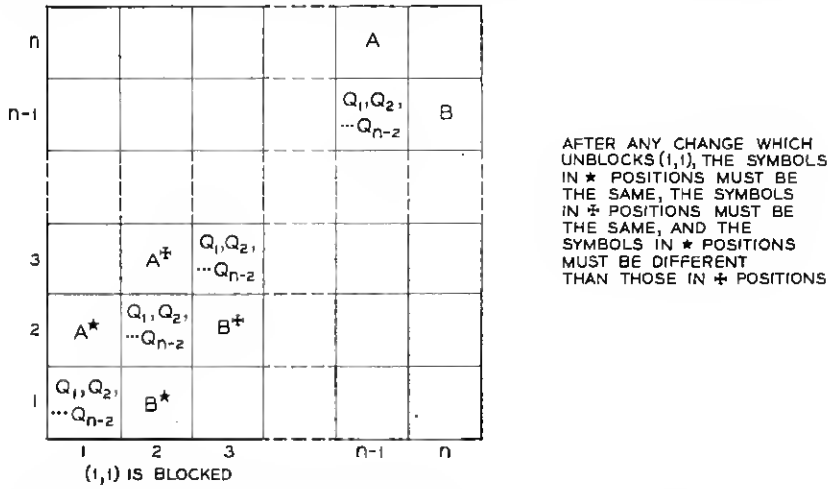

Fig. 3 — Matrix representation of the blocked network state which requires a maximum of changes to be unblocked.

there were no such symbol, then every symbol in column c_1 would also appear in row r_1. And since all n symbols must appear in the union of column c_1 and row r_1 (condition for non-trivial blocking), it would follow that all n symbols appear in column c_1 making (c_1, r_1) trivially blocked, a contradiction of our hypothesis. Similarly *there must be a symbol, say B, in row r_1 which does not appear in column c_1*.

Let A be in (r_2, c_1).

Let B be in (r_1, c_2).

Thus far we have completely defined:

r_1, the row in which the blocked connection appears,

c_1, the column in which the blocked connection appears,

r_2, the row in which the A in column c_1 appears, and

c_2, the column in which the B in row r_1 appears.

Now we wish to define other rows and columns:

r_3, the row in which an A appears in column c_2 if there is such a row (otherwise r_3 is undefined),

c_3, the column in which a B appears in row r_2 if there is such a column (otherwise c_3 is undefined),

r_4, the row in which an A appears in column c_3 if c_3 is defined and there is such a row (otherwise r_4 is undefined),

c_4, the column in which a B appears in row r_3 if r_3 is defined and there is such a column (otherwise c_4 is undefined).

In general, for all $j > 1$:

r_j is defined to be the row in which A appears in column c_{j-1}, provided c_{j-1} is defined, and provided that A does appear in column c_{j-1}. If not, r_j is undefined.

c_j is defined to be a column in which B appears in row r_{j-1}, provided r_{j-1} is defined, and provided that B does appear in row r_{j-1}. If not, c_j is not defined.

The above definition has the important property that if r_j and r_k are both defined, and $j \neq k$, then $r_j \neq r_k$. Also, if c_j and c_k are both defined, and $j \neq k$, then $c_j \neq c_k$. This is justified by the following argument: consider the sequence

$$r_1, c_1, r_2, c_2, \cdots r_i, c_i, \cdots r_n, c_n. \tag{1}$$

Assume there is a first member equal to a previous member of the sequence.

This is either a row or column.

1. Assume row r_j is the first member of the sequence which is both defined and the same as a previous defined member, say r_k, $k \neq j$. First of all k cannot be 1 since r_j, $j > 1$ is defined to have an A in it, and row r_1 has no A in it. So $r_1 \neq r_j$, $j > 1$. Now then assume $k > 1$, $j > 1$,

$j \neq k$. Then an A appears in (r_k , c_{k-1}), and in (r_j , c_{j-1}), (by our definition of r_j). So unless $c_{k-1} = c_{j-1}$, $j - 1 \neq k - 1$, there would be two different A's in row $r_k = r_j$. There cannot be two different A's in a single row. Therefore $c_{k-1} = c_{j-1}$. But this contradicts the assumption that the first member having this property is row r_j. That leaves only the possibility of column c_j being the first such member.

2. Then assume column c_j is the first member of the sequence which is both defined and the same as a previous defined member, say column c_k $k \neq j$. Then $k \neq 1$, because c_1 has no B, and c_j, $j > 1$ does by definition of c_j. If $k > 1$, $j > 1$, $k \neq j$ and $c_j = c_k$, then for similar reasons to those of the above paragraph $r_{j-1} = r_{k-1}$. Therefore our second assumption, $c_j = c_k$, is also contradicted, completing the proof.

Having shown that the defined members of (1) are distinct, we wish now to examine this sequence further. For convenience it is rewritten below.

$$r_1 , c_1 , r_2 , \cdots r_j , c_j , \cdots r_n , c_n \tag{1}$$

There is a first member of this sequence which is defined but whose succeeding member is undefined, say c_f. Since r_{f+1} is the first member of the sequence which is undefined, it follows from the definition of r_j that there is no A in column c_f. We then know that according to our definition of c_j and r_j the following matrix positions contain A's

$$(r_2 , c_1); (r_3 , c_2); \cdots (r_f , c_{f-1})$$

and the following contain B's

$$(r_1 , c_2); (r_2 , c_3); \cdots (r_{f-1} , c_f)$$

Now in order to unblock (r_1 , c_1) we make the following changes.

2.3.2 *Change Algorithm*

Change the original B's to A's in columns c_j:$j = 3, 5 \cdots f$ if f is odd (or in columns $j = 2, 4 \cdots$ if f is even). This involves changing B's to A's in rows r_j: $j = 2, 4 \cdots f - 1$ if f is odd ($j = 1, 3 \cdots f - 1$ if f is even).

Change the original A's to B's in rows r_j, $j = 2, 4 \cdots f - 1$, if f is odd ($j = 3, 5 \cdots f - 1$ if f is even). This involves changing A's to B's in columns c_j: $j = 1, 3, \cdots f - 2$, if f is odd ($j = 2, 4 \cdots f - 2$, if f is even). *Note that the total number of changes is* $f - 1$.

We see that if f is odd then after the change (r_1 , c_2) will still contain a B, but (r_2 , c_1) which formerly contained an A now contains a B. Therefore an A may now be legitimately placed in (r_1 , c_1). A similar argument holds if f is even.

It remains to show that the changes we have prescribed do not lead to any conflicts. For this demonstration assume f is odd. (A similar argument holds for f even.) The only conflicts possible must involve A's and B's, since these are the only symbols changed and resulting from the change. Furthermore, the only conflicts possible are in rows r_1, $r_2 \cdots r_f$ or columns c_1, $c_2 \cdots c_f$, since these are the only rows and columns in which changes were made. Also, at most one A has been added to any row or any column. Similarly, at most one B has been added to any row or column. Before the change there were single A's in columns c_j, $j = 1$ to $f - 1$, and in rows r_j, $j = 2$ to f. As a result of the change single A's were added to columns c_i, $i = 3, 5 \cdots f$ and no others, and to rows r_i, $i = 2, 4 \cdots f - 1$ and no others. So it is only these columns and rows which could possibly contain more than one A. But these columns and rows each contain only a single A, because although an A has been added to each, the original A in each has been changed to a B. For according to our prescribed changes, the original A's in c_i: $i = 1, 3 \cdots f - 2$ were changed to B's. This takes care of all columns to which an A was added except column c_f, and column c_f did not originally contain an A. Also, the original A's in rows r_i: $i = 2, 4 \cdots f - 1$ were changed to B's and this takes care of all rows to which an A was added.

Again as a result of the change, single B's were added to rows r_i: $i = 2, 4, 6, \cdots f - 1$, and to columns c_i: $i = 1, 3, 5 \cdots f - 2$. It is therefore only those columns and rows which could have more than one B. But the original B's in columns c_j, $j = 3, 5 \cdots f$ have been changed to A's. This takes care of all columns to which a B was added except column c_1, and column c_1 originally did not have a B. Also, the original B's in rows r_i: $i = 2, 4 \cdots f - 1$ were changed A's and this takes care of all rows to which a B was added.

If all members of sequence 1 are defined, then c_n is the last defined member, and there cannot be an A in c_n, because such an A would have to be in some row other than row r_1. There are A's in all rows other than r_1, but none of these A's are in c_n. This follows from the definition of r_j. From here, then, our argument goes on as the general case in which r_{f+1} was the first undefined member of sequence (1).

Thus the maximum number of changes required to unblock a call is $n - 1$.

2.3.3 *Necessity*

The network has n intermediate switches which we represent by the symbols A, B, $Q_1 \cdots Q_{n-2}$. Assume that (1, 1) is blocked by the follow-

ing network state:

(i, i); $i = 1$ to $n - 1$ each contain all the symbols $Q_1, \cdots Q_{n-2}$.

$(i, i + 1)$; $i = 1$ to $n - 1$ each contain the symbol B.

$(i + 1, i)$; $i = 1$ to $n - 1$ each contain the symbol A.

There are no other symbols in the matrix. To unblock (1, 1) the symbols in (1, 2) and (2, 1) must be made the same because:

(a) After any change there must still be $n - 2$ different symbols in (1, 1)

(b) There must still be one symbol in (1, 2) different from all those in (1, 1)

(c) There must still be one symbol in (2, 1) different from all those in (1, 1)

(d) If then the symbols in (1, 2) and (2, 1) were different, there would be a total of n symbols in row 1 and column 1, leaving no symbol available to unblock (1, 1).

Assume that the symbols in $(i + 1, i)$ and $(i, i + 1)$; $i = k - 1$ must be the same, say X, in order to unblock (1, 1). Now (i, i); $i = k$, which is in row k must, after the change, still contain $n - 2$ different symbols, say $Y_1, Y_2 \cdots Y_{n-2}$. The symbol X in $(i + 1, i)$; $i = k - 1$ which is also in row k must be different from $Y_1, Y_2 \cdots Y_{n-2}$. Therefore the symbol in $(i, i + 1)$; $i = k$ which is also in row k must be different than $X, Y_1, Y_2 \cdots Y_{n-2}$. There is only one symbol that can be different from all $n - 1$ different symbols $X, Y_1 \cdots Y_{n-2}$, say Z. So Z must appear in $(i, i + 1)$, $i = k + 1$. Similarly as stated previously (i, i), $i = k$, which is in column k, must still have the $n - 2$ different symbols $Y_1, Y_2 \cdots Y_{n-2}$. Also in column k the symbol X is in position $(i, i + i)$, $i = k - 1$. Therefore it follows that the symbol in $(i + 1, i)$, $i = k$, which is also in column k must be different than $X, Y_1, \cdots Y_{n-2}$, and must be Z.

Hence the induction is complete, proving that if (i, i) $i = 1$ to $n - 1$ each contain $n - 2$ different symbols (this must be true because of the given initial network state), and (1, 1) is to be unblocked, then for each $i = 1$ to $n - 1$, the pair $(i + 1, i)$ and $(i, i + 1)$ must contain the same symbol. Since initially $(i + 1, i)$ contained a different symbol from $(i, i + 1)$ for $i = 1$ to $n - 1$, at *least* $n - 1$ *changes are necessary* to put the network in a state both equivalent to its initial state, and in which (1, 1) is unblocked.

III. COMPARISON WITH SLEPIAN'S RESULT

I have been able to obtain the bound of $n - 1$ on the number of changes by considering changes of both input and output connections involved in the blocked connection, that is changes in both rows and

columns of our matrix. Slepian, on the other hand, considered, in effect, only the changes in rows. That is, he gave a method for changing the blocking symbol in a row (B in our proof) without taking advantage of the symbol in the column (A in our proof) for reducing the number of changes.

In the following sections a number of generalizations are developed.

IV. METHODS FOR RESWITCHING A NETWORK TO UNBLOCK CALLS

In the proof of Theorem 1 there is a method given for determining the changes required to unblock a blocked connection. This method involves two second-stage switches (A and B in the proof). If (r_1, c_1) is blocked, we look for a symbol in r_1 not in c_1, and a symbol in c_1 not in r_1, and carry out the Change Algorithm of Theorem 1 (Section 2.3.2). We will call this "method 1." We could use a slightly more complex method in which we test all symbol pairs, (A, B), such that A is in r_1 but not in c_1, and B is in c_1 but not in r_1, to find which pair will require the fewest changes. The changes are then made on this pair according to the change Algorithm. We will call this "method 2." Methods 1 and 2 both involve changing only two second-stage switches. We can develop methods which are not restricted to changes of only two second-stage switches.

Assume that (r_1, c_1) is blocked. Assume that A is in r_1, but not in c_1 and B is in c_1 but not in r_1. As in the proof of the theorem, assume c_f is the first member of sequence (1), which is itself defined but whose succeeding member is not defined. Then by making $f - 1$ changes of A's and B's, we know (r_1, c_1) could be unblocked. If, however, some of the A's or B's which serve to define c_j, and r_j, $j < f$, could be changed *without conflict* to a symbol other than A or B, to C for example, then we could unblock (r_1, c_1) in less than $f - 1$ changes. This is best illustrated by an example (see Fig. 4). In summary, method 3 involves: first, finding a symbol in r_1 not in c_1, say A; a symbol in c_1, not in r_1, say B; second, finding the last defined term of sequence (1); and third, examining the A's and B's which define sequence (1) to determine if any can be changed to a symbol other than A or B. If not, change the B's to A's according to the Change Algorithm. If so (say a B in column k, $k < f$, can be changed to a C without conflict) then make this change and make all changes given by the change algorithm in columns c_j, for $j < k$ rows r_j for $j < k$.

In method 4 we try method 3 on all pairs of symbols (A, B); A in r_1 not in c_1; B in c_1 not in r_1, and actually carry it out on the pair which requires the fewest changes.

The methods discussed vary in complexity. A legitimate question to

ask is, what do we get for this complexity? The least upper bound on the number of changes required to unblock a blocked call has been established and is independent of which of the four methods is used. The greater complexity, however, does serve to decrease the average number of changes required per blocked call. We can get a quantitative idea of the value of the different methods by finding for any number of changes required to unblock a call, a lower bound on the number of calls which must already be set up in the network for each of the four methods.

These bounds are illustrated by examples in which it can be seen that the removal of any call will lower the number of changes required. These can be shown to be greatest lower bounds (Fig. 5)

x = the number of changes required

y = the number of calls already set up.

For methods 1 and 2

$$y = 2x + n - 2 \quad \ldots \text{bound 1 (See Fig. 5a).}$$

For methods 3 and 4

$$y = 2x + \frac{x}{2}(n - 2) \qquad \text{for } x \text{ even}$$

$$y = 2x + \frac{x + 1}{2}(n - 2) \qquad \text{for } x \text{ odd} \qquad \cdots \text{ bound 2 (See Fig. 5b).}$$

These bounds do not indicate the difference between methods 1 and 2, or between methods 3 and 4, because as far as these bounds are concerned there is no difference. However, Fig. 6 indicates a case in which

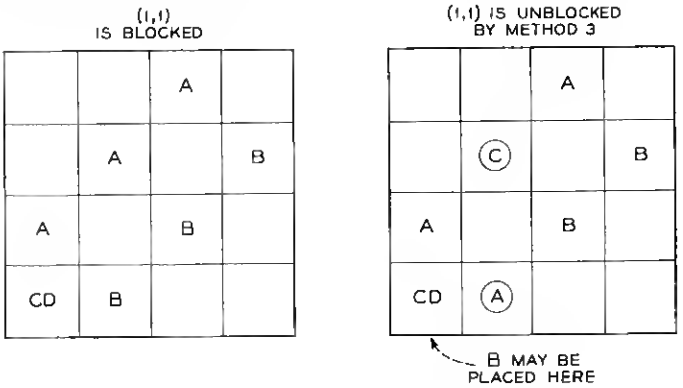

Fig. 4 — Illustration of change method 3.

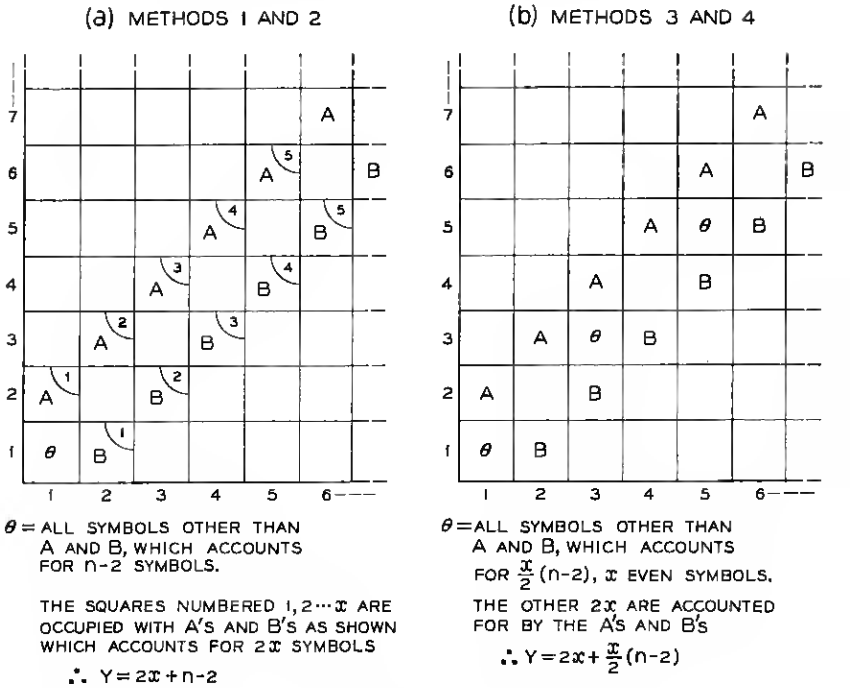

Fig. 5 — Illustrations of the smallest number of calls which must occupy a network (x) if a call is blocked and x changes are required to unblock the call.

method 2 or 4 requires (assuming A and B were changed) one change to unblock (1, 1), whereas methods 1 and 3 require five.

There are more complex methods possible, in which more complex changes are allowable than any of the four methods, discussed above. Fig. 7 indicates how a network state which requires four changes with method 4 could be unblocked with three changes. It would be interesting to obtain the general bound, equivalent to bounds 1 and 2 in the case in which no restriction was made on possible changes.

A 704 program for simulating method 2 on a simulated four-stage network is being written by J. Nervik. Also it should be fairly simple to realize circuitry for any of the four methods described.

V. GENERALIZATION TO MORE THAN THREE STAGES

In the proof of Theorem 1 we were able to show how the necessary changes to unblock a blocked connection can be made disturbing at most only two second-stage switches. This gives the following corollary.

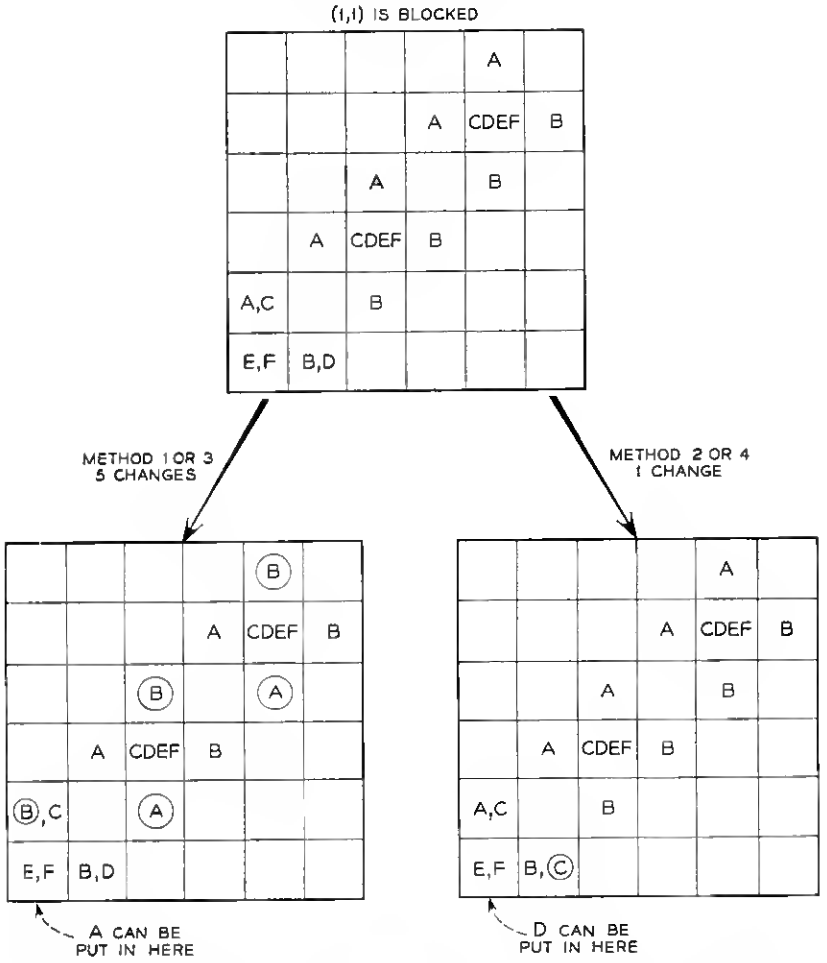

Fig. 6 — Illustration of the advantage of methods 2 and 4 over methods 1 and 3 respectively.

5.1 *Corollary* 1

A blocked connection may be unblocked by changing connections in such a way as to disturb no more than two second-stage switches.

This corollary will serve to obtain bounds on the number of calls which are disturbed in unblocking a blocked connection for five, seven, and nine-stage networks.

The five-stage network to which we refer is obtained by starting with

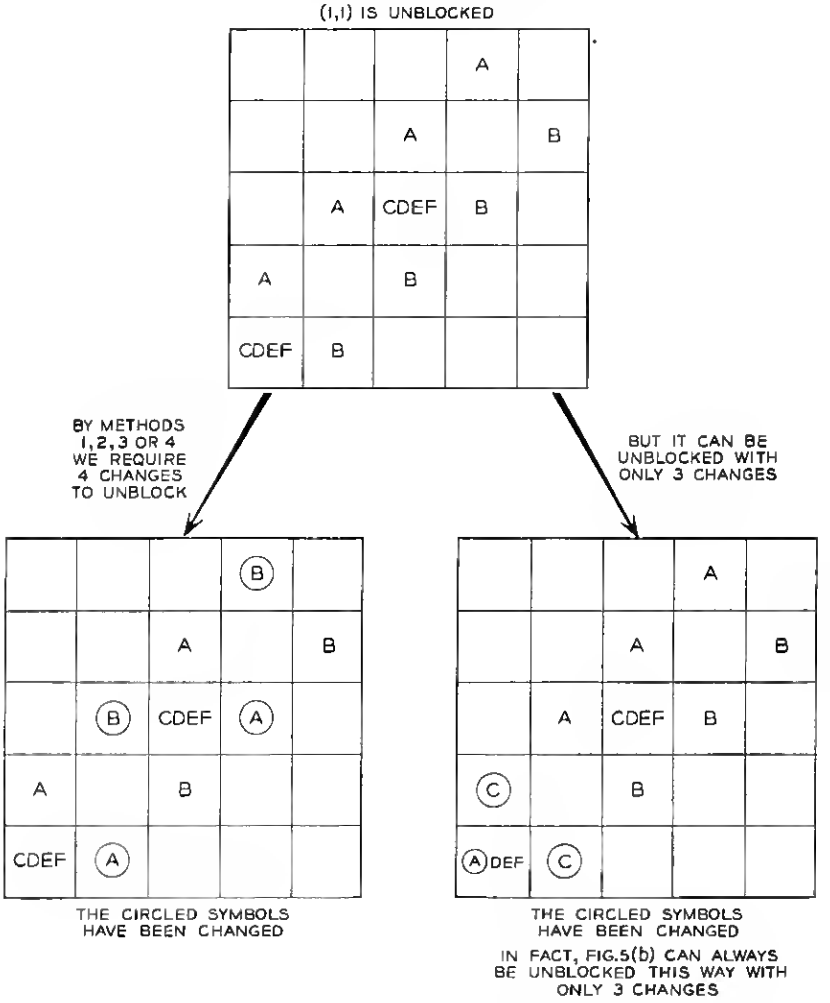

Fig. 7 — A change method more efficient than method 4.

a three-stage network as in Fig. 1. Each second-stage switch of this network is then expanded into a three-stage switch having $\sqrt{n}$ input 1st, 2nd, and 3rd stage switches (Fig. 8).

Now suppose a three-stage switch is blocked. We can find the two switches (say A and B) in which changes can be made by the Change Algorithm. We calculate the changes which must be made in A and B by the same algorithm. When we are done we have a list of all the connections which must finally be made in second stage switches A and B. Now

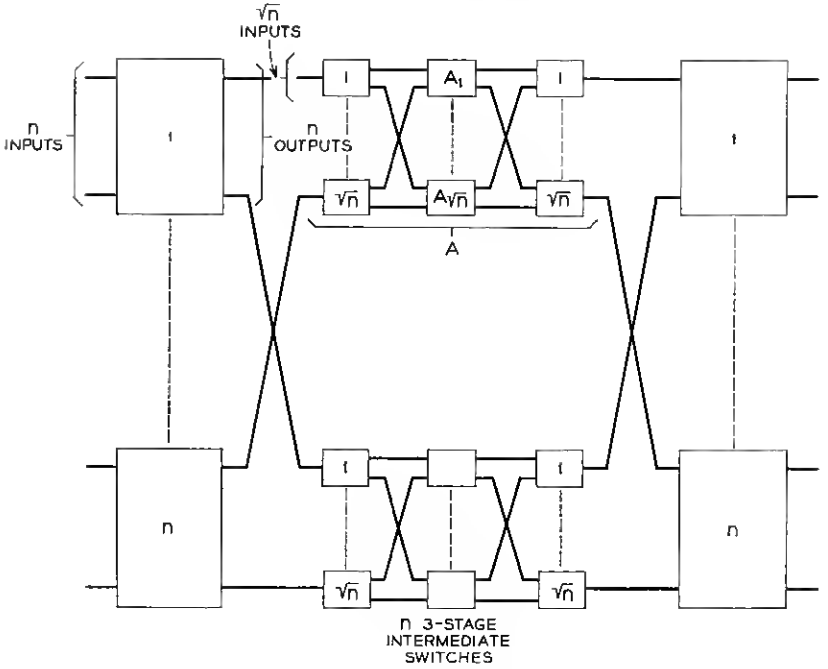

Fig. 8 — Five-stage network suitable for reswitching.

all connections which are initially set up in switches A and B may be taken down, and the new set of connections put up in their place.

If A and B are the two switches which are changed by the Change Algorithm, there can be no more than $n - 1$ connections in either of them, since by definition there is no A in c_1, and no B in r_1, assuming (r_1, c_1) is blocked. Since we take down all connections in A and B, no more than $2n - 2$ calls are disturbed.

If A and B were three-stage switches themselves, our network and the above argument would remain the same. The only question which might arise is whether we could make the final connections in three-stage switches A and B. These connections could be made. One would calculate exactly how to set them up by repeated application of the Change Algorithm. First one would conceptually set up one connection arbitrarily, then one would try to conceptually set up the next connection. If it were blocked, the Change Algorithm could be used to unblock it. This would continue until the final connections were decided conceptually. Then they could actually be made.

This result extends easily to seven, nine $\cdots$ stage switches.

5.2 *Generalization of Theorem* 1

For a q-stage network, q odd and $q > 3$, *of the type described above (Fig. 8), no more than* $2n - 2$ *calls need be disturbed to unblock a blocked call.*

This bound can probably be lowered. If A and B are the two second-stage switches in which connections are to be changed, it has been shown that no more than a total of $n - 1$ connections in both A and B need be changed. However, if A and B are themselves three-stage switches, in order to make the initial $n - 1$ changes it may be necessary to juggle other connections in both A and B. A closer study of the ways in which this juggling can be done might serve to lower the $2n - 2$ bound.

5.3 *Generalizations to Other Network Configurations*

So far we have discussed networks in which all stages have the same number of switches. The matrix representation, with the restrictions given in Section 2.1, is applicable to a more general class of three-stage networks than that of Fig. 1. In this more general class the numbers of

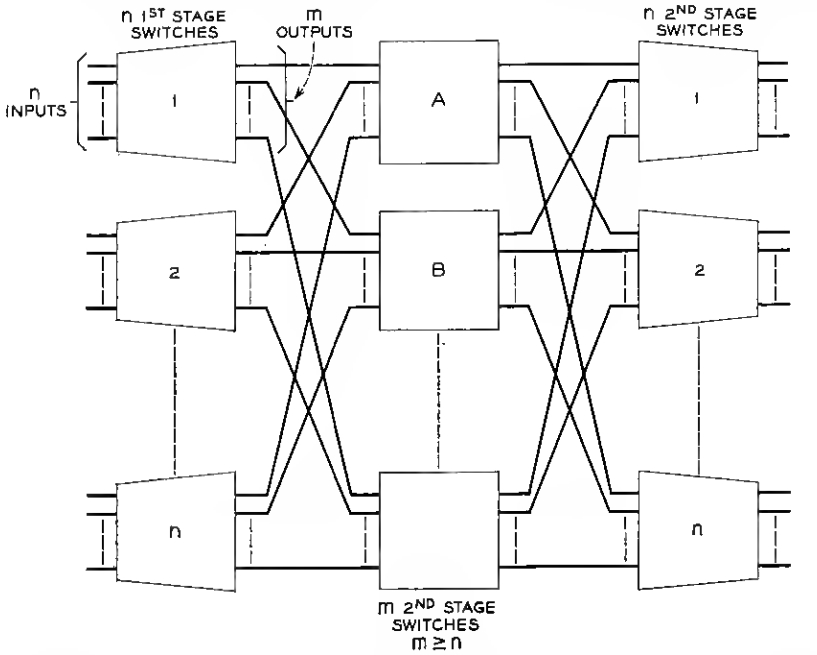

Fig. 9 — Generalization to more second-stage switches than first-stage switches.

intermediate switches may be greater than n (Fig. 9). Assume there are m intermediate switches, $m > n$. There are still n input (output) switches. There is one link from each input (output) switch to each of the intermediate switches.

For this more general class of switches the condition for legitimate blocking (Section 2.1) must be generalized to read:

(a, b) is legitimately blocked if there are a total of m different symbols in row a and column b. There are less than n symbols in row a (column b).

We have shown that if $m = n$, then no more than $n - 1$ changes are required to unblock a blocked connection. Clos[2] has shown that if $m = 2n - 1$, the network is nonblocking. (0 changes are required to unblock a blocked connection.) I can also prove that if $m = 2n - 2$, no more than one change is required to unblock a blocked connection. (This is justified later.) These results lead to the conjecture *that if* $m = 2n - j$, *no more than* $j - 1$ *changes are required to unblock a blocked connection.*

We will now prove a simple lemma which was useful in finding the bounds for $m = 2n - 2$, and which may prove helpful in attacking our conjecture.

5.3.1 *Lemma*

If $m = n + k$ *and* (a, b) *is legitimately blocked, then there must be at least* $k + 1$ *symbols in row* a, *none of which are in column* b, *and there must be at least* $k + 1$ *symbols in column* b, *none of which are in row* a.

5.3.2 *Proof*

By assumption, (a, b) is blocked; therefore there are a total of $n + k$ different symbols in row a and column b, by the blocking condition. There are less than n symbols in row a, and less than n symbols in column b, also by the blocking condition.

Let R = no. of symbols in row a, not in column b
C = no. of symbols in column b, not in row a
X = no. of symbols appearing in (a, b)
B = no. of symbols in both row a, and column b, but not (a, b)

Then we have

1. $B + C + B + X = n + K$
2. $R + B + X \quad < n$
3. $C + B + X \quad < n$

which by substituting for X in 2, and 3 the value obtained from 1 gives:

$$C < k$$
$$R < k$$

We illustrate the use of this lemma by proving the Clos non-blocking network is non-blocking. In the Clos network $k = n - 1$. Therefore if (a, b) is legitimately blocked, there must be $n = k + 1$ symbols in row a $(C = n)$. This immediately contradicts the hypothesis that (a, b) is legitimately blocked.

For the case in which $k = n - 2$, it follows from our lemma that if (a, b) is blocked, $C = n - 1$ and $R = n - 1$. If this were all the connections that were up, a single change of any of the C symbols in row a (called a-symbols) or any of the R symbols in column b (called b-symbols) would unblock (a, b). So, in order that more than one change will be required, all the proposed unblocking changes must produce conflicts. This means that in the column of each of the symbols in row a, all $n - 1$ b-symbols must appear. Also in the row of each of the symbols in column b, all $n - 1$ a-symbols must appear. It follows that if there are to be no more than n symbols in any row or column that there must be no symbol in (a, b), and one symbol in every other location in row a, and column b. Now we look at row k. There must be one b-symbol and all $n - 1$ a-symbols in this row. One of the a-symbols in row k must be in column p, $(p \neq a)$. In column p however, there must already be an a symbol and $n - 1$ b-symbols, these together with the a-symbol in row k, column p total to $n + 1$ symbols in column p. This is not allowed. Therefore one change will always be sufficient to unblock a blocked connection of $m = 2n - 2$.

VI. CONCLUSION

There are other directions in which generalizations appear feasible with the techniques of this paper. We can deal with rectangular matrices in an analogous manner to that used for the square matrix here. These correspond to concentration networks. Triangular networks seem somewhat more difficult, but still feasible to treat. Finally, results on various network configurations can probably be generalized to more than three stages.

We have discussed here the use of the reswitching to make networks non-blocking. One might also consider a more modest goal in which provision is made for fewer than the number of reswitches or changes required to make the uetwork non-blocking, in an attempt to improve blocking characteristics. The program being written by J. Nervik will be used to obtain some estimate of this improvement.

APPENDIX

A.1 *Introduction*

In the body of the paper there are some algorithms by which networks not originally non-blocking can be made non-blocking by rearranging connections already set up in the network. These algorithms involve the temporary disturbance of calls already set up in the network.

Here I propose to describe a slight modification of the network and of the algorithm which will allow one to make the network essentially non-blocking or rearrangable without requiring any disturbance of existing calls set up in the network.

A.2 *Network Modification*

The basic three-stage network, each stage requiring n, $n \times n$ switches, is modified to a three-stage network in which stage 1 consists of n, $(n) \times (n + 1)$ switches, stage 2 consists of $n + 1$, $(n) \times (n)$ switches, stage 3 consists of n, $(n) \times (n + 1)$ switches. Each first (third) stage switch has one link to each second-stage switch, as pictured in Fig. 10.

As in the body, we represent the connections in this network with a $n \times n$ matrix. For each input switch, there is a row in the matrix. These are numbered 1 to n. For each output switch, there is a column in the matrix. These are numbered 1 to n. A connection between input switch j and output switch k through middle switch A is indicated by an A in position (j, k). Middle switches are lettered A, B, etc. There are $n + 1$ letters. There cannot be more than $n + 1$ letters* in any row or column or location of the matrix.

A.3 *Algorithm Modification*

In order to make this network essentially non-blocking we use the following procedure.

We choose not to use middle switch A until we get a blocked condition. When we get a blocked condition, we are in the same situation as if the network were a three-stage network with each stage having n $n \times n$ switches. From the corollary of Section V, we know that this blocked connection could he unblocked without disturbing more than two (intermediate) middle switches (not including switch A, which has not as yet been used). Suppose the two middle switches in which connections are to be changed to unblock the blocked connection are C and D. According to the change algorithm we would change a certain set of connections

* In the operation of this network there are times during which a single input lead is connected to two middle switches. Thus, although there are only n inputs per input switch there may be $n + 1$ connections in a single input switch.

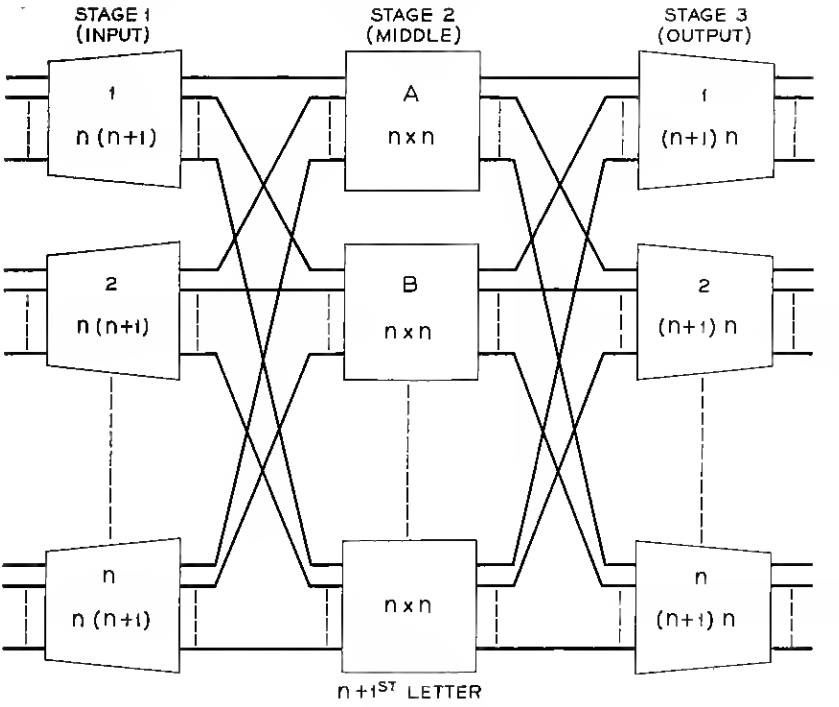

Fig. 10 — Modified three-stage network.

in middle switch C to connections in middle switch D, and a certain set of connections in middle switch D to connections in middle switch C. This would leave either switch C or D available for use for the blocked call. (Which particular switch was available depends on the exact choice of the sets of connections in C and D which are to be rearranged.) Such a rearrangement involves disturbing all the calls using the set of connections in C and D which are to be changed. In the modified network we have an extra middle switch A available which we can use to maintain all calls while connections are being rearranged. The modified algorithm is given below. The steps of the algorithm are illustrated by an example in Figure 11.

According to the algorithm, we find the set of connections in middle switches C and D which must be rearranged to unblock the blocked connections.

1. For every connection in C which we have elected to change, we add a corresponding connection in A. In the matrix representation this

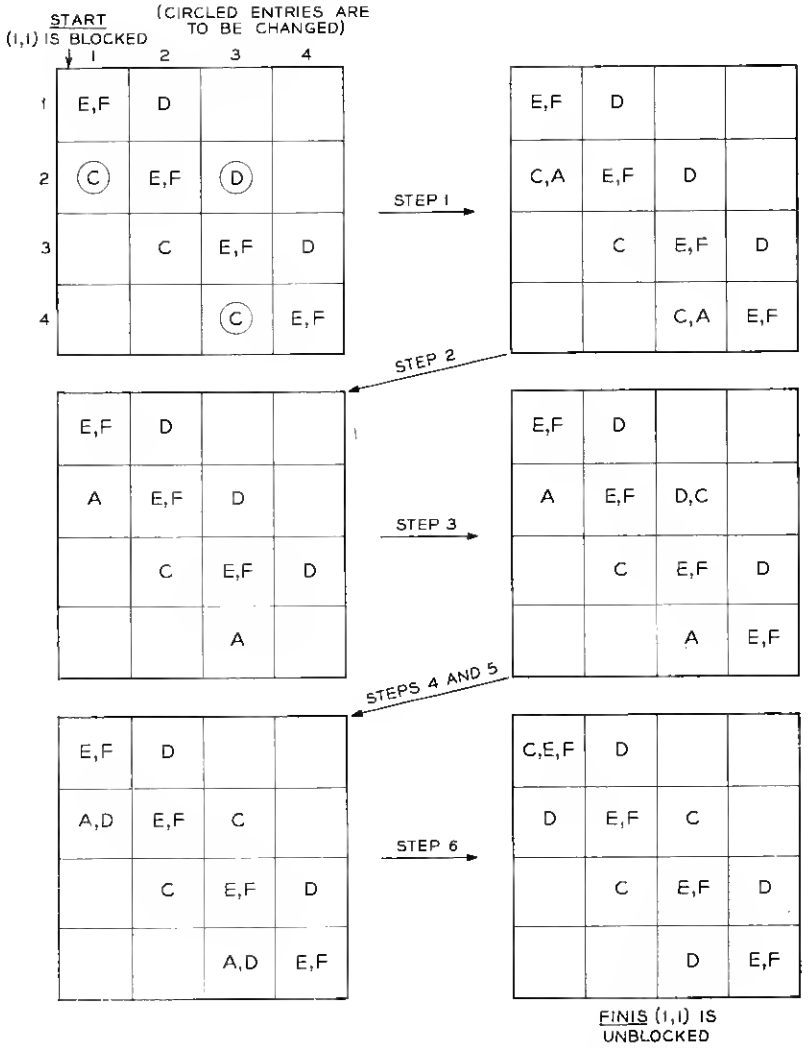

Fig. 11 — Example of rearranging without disturbing.

corresponds to adding an A in every position in which an elected C appears.

2. All elected C connections are taken down; the calls originally carried by these connections are now carried by switch A.

3. For every connection in D which we originally elected to change,

we add a corresponding connection in C. This means in the matrix representation an addition of a C in every position in which there is an elected D. This will always be possible because we have already taken down all the elected C connections which, according to the body of the paper, are the only connections which would prevent this addition.

4. All elected D connections are taken down; the calls carried by these D connections are now carried by C connections.

5. To every connection carried by A we add a corresponding connection carried by D. This will always be possible according to the results in the body of the paper.

6. Finally, we take down all connections in A.

We thus have carried out the change algorithm, and therefore, have unblocked the blocked connection. Switch A has no connection set up in it so it is available for use in unblocking the next blocked call.

A.4 *Making Use of Dead Time* (Time during which there is no activity in the network)

Actually, the blocked call is unblocked after step 2, at which time the desired connection can be put up using switch C (assuming that the C in the row or column of the blocked call was changed in step 1 and step 2). Since either a C or D in the row or column of the blocked call is changed in this algorithm the original choice of where to add connections in A could be made so that the call would be unblocked after step 2.

Steps 1 and 2 could have been combined in such a way as to add a connection in A, take down the corresponding connection in C, then add another connection in A, take down the corresponding connection in C, and so forth. In this case the blocked connection could be unblocked after the first addition of a connection in A and the removal of its corresponding connection in C.

If the single switch A is to serve to allow for unblocking calls without disturbing other calls, by our algorithm, A must be completely available when a blocked connection is to be unblocked. So although the blocked call may be unblocked early in the algorithm, the remainder of the algorithm must have been completed before the next blocked call is to be unblocked. The extra switch A acts as a kind of connection memory so that normal dead time in the network may be profitably used for improving blocking characteristics.

By adding additional middle switches analogously to the way A was added, we would effectively add more connection memory and increase the efficient use of dead time in the network. Thus, if there were two additional switches A and B, A would not have to be cleared before

another blocked call could be handled because B would be available. A would have to be cleared before the second blocked call after the one that engaged A were encountered.

There are some indications that the simple scheme proposed here can be improved upon.

In this scheme we are calling on A to handle no more than $n/2$ connections at any one time, and A is in use at all only during the unblocking operation. In the network A appears like any other middle-stage switch, but its function is much different from the other middle-stage switches. Perhaps the organization could be changed to share the load more symmetrically.

REFERENCES

1. Slepian, D., unpublished memorandum, March 25, 1952.
2. Clos, C., B.S.T.J., **32**, March, 1953, pp. 406–424.